



## Cambridge O Level

---

COMPUTER SCIENCE

2210/21

Paper 2

May/June 2021

MARK SCHEME

Maximum Mark: 50

---

**Published**

This mark scheme is published as an aid to teachers and candidates, to indicate the requirements of the examination. It shows the basis on which Examiners were instructed to award marks. It does not indicate the details of the discussions that took place at an Examiners' meeting before marking began, which would have considered the acceptability of alternative answers.

Mark schemes should be read in conjunction with the question paper and the Principal Examiner Report for Teachers.

Cambridge International will not enter into discussions about these mark schemes.

Cambridge International is publishing the mark schemes for the May/June 2021 series for most Cambridge IGCSE™, Cambridge International A and AS Level components and some Cambridge O Level components.

---

This document consists of **11** printed pages.

**PUBLISHED****Generic Marking Principles**

These general marking principles must be applied by all examiners when marking candidate answers. They should be applied alongside the specific content of the mark scheme or generic level descriptors for a question. Each question paper and mark scheme will also comply with these marking principles.

**GENERIC MARKING PRINCIPLE 1:**

Marks must be awarded in line with:

- the specific content of the mark scheme or the generic level descriptors for the question
- the specific skills defined in the mark scheme or in the generic level descriptors for the question
- the standard of response required by a candidate as exemplified by the standardisation scripts.

**GENERIC MARKING PRINCIPLE 2:**

Marks awarded are always **whole marks** (not half marks, or other fractions).

**GENERIC MARKING PRINCIPLE 3:**

Marks must be awarded **positively**:

- marks are awarded for correct/valid answers, as defined in the mark scheme. However, credit is given for valid answers which go beyond the scope of the syllabus and mark scheme, referring to your Team Leader as appropriate
- marks are awarded when candidates clearly demonstrate what they know and can do
- marks are not deducted for errors
- marks are not deducted for omissions
- answers should only be judged on the quality of spelling, punctuation and grammar when these features are specifically assessed by the question as indicated by the mark scheme. The meaning, however, should be unambiguous.

**GENERIC MARKING PRINCIPLE 4:**

Rules must be applied consistently, e.g. in situations where candidates have not followed instructions or in the application of generic level descriptors.

**GENERIC MARKING PRINCIPLE 5:**

Marks should be awarded using the full range of marks defined in the mark scheme for the question (however; the use of the full mark range may be limited according to the quality of the candidate responses seen).

**GENERIC MARKING PRINCIPLE 6:**

Marks awarded are based solely on the requirements as defined in the mark scheme. Marks should not be awarded with grade thresholds or grade descriptors in mind.

**Please note the following further points:**

The words in **bold** in the mark scheme are important text that needs to be present, or some notion of it needs to be present. It does not have to be the exact word, but something close to the meaning.

If a word is underlined, this **exact** word must be present.

A single forward slash means this is an alternative word. A double forward slash means that this is an alternative mark point.

Ellipsis (...) on the end of one-mark point and the start of the next means that the candidate **cannot** get the second mark point without being awarded the first one. If a MP has ellipsis at the beginning, but there is no ellipsis on the MP before it, then this is just a follow-on sentence and **can** be awarded **without** the previous mark point.

**PUBLISHED**

Question	Answer	Marks
<b>Section A</b>		
1(a)(i)	<p>Many correct answers. They must be meaningful and related to <b>Task 1</b>. The names are examples only.</p> <p><b>One</b> mark per mark point</p> <ul style="list-style-type: none"> <li>• Constant    <code>MaxCandidates</code></li> <li>• Value        <code>4</code></li> <li>• Use            The value of the maximum number of candidates for the election</li> </ul>	<b>3</b>
1(a)(ii)	<p>Many correct answers. They must be meaningful and related to <b>Task 1</b>. The names are examples only.</p> <p><b>One</b> mark per mark point</p> <ul style="list-style-type: none"> <li>• Variable    <code>NumberCandidates</code></li> <li>• Use            Storing the number of candidates in the election (for a tutor group)</li>   <li>• Array        <code>CandidateNames</code></li> <li>• Use            Storing the names of the candidates for the election</li> </ul>	<b>4</b>
1(b)	<p><b>One</b> mark per mark point (<b>Max 4</b>)</p> <p>MP1    Change the value of the <code>MaxCandidates</code> constant/variable to 8</p> <p>MP2    Change the input message to state the maximum number of candidates is 8 ...</p> <p>MP3    ...how your program changed the input message</p> <p>MP4    Change the loop limit to up to 8 ...</p> <p>MP5    ...how your program changed the loop limit</p> <p>MP6    Change the validation to allow input up to 8 ...</p> <p>MP7    ...how your program changed its validation check</p> <p>MP8    Change the array size(s) to ensure sufficient capacity to store up to 8 names ...</p> <p>MP9    ...how your program changed the array sizes</p> <p>MP10   Change the counters to ensure votes can be counted for up to 8 candidates ...</p> <p>MP11   ...how your program changed its counters</p>	<b>4</b>

Question	Answer	Marks
1(c)	<p>Any <b>five</b> from:</p> <p>MP1 Input with message to enter unique voter number</p> <p>MP2 Validation of (unique) voter number entered e.g. length check/type check/range check</p> <p>MP3 Attempt to check if voter number input is in list of possible voters</p> <p>MP4 Attempt to check if they have already voted</p> <p>MP5 If voter has already voted, message to warn them they can't vote</p> <p>MP6 Attempt at preventing them from voting</p> <p>MP7 Store voter number in a suitable data structure</p> <p><b>Example answer</b></p> <pre> OUTPUT "Please enter your unique voter number" INPUT UniqueVoterNumber FoundFlag ← False AllNumbersChecked ← False Counter ← 0 WHILE FoundFlag = False AND AllNumbersChecked = False   IF StudentNumbers[Counter] = ""     THEN       AllNumbersChecked = True       StudentNumbers[Counter] ← UniqueVoterNumber     ELSE       IF UniqueVoterNumber = StudentNumbers[Counter]         THEN           FoundFlag = True           PRINT "Sorry, you have already voted"         ELSE           Counter = Counter + 1         ENDIF       ENDIF     ENDWHILE   IF FoundFlag = False     THEN       OUTPUT "Please enter the code of your chosen candidate"       INPUT Vote     ENDIF </pre>	<b>5</b>

**PUBLISHED**

<b>Question</b>	<b>Answer</b>	<b>Marks</b>
1(d)	Explanation of how the program does the following: Any <b>four</b> from: MP1 Find out how many votes in total (for all candidates) were cast in the election. MP2 For each candidate MP3 ... calculate the percentage of votes MP4 ... excluding abstentions. MP5 Display the name of each candidate, the number of votes and the percentage of votes they received with appropriate messages. MP6 Display the number of votes cast and the number of abstentions with appropriate message.	<b>4</b>

Question	Answer				Marks
<b>Section B</b>					
2	<b>One</b> mark per correct column				<b>3</b>
	<b>Statement</b>	<b>Validation</b>	<b>Verification</b>	<b>Both</b>	
	Entering the data twice to check if both entries are the same.		✓		
	Automatically checking that only numeric data has been entered.	✓			
	Checking data entered into a computer system before it is stored or processed.			✓	
Visually checking that no errors have been introduced during data entry.		✓			

Question	Answer		Marks
3	<b>One</b> mark per bullet point		<b>6</b>
	37		
	• Data type name	Integer	
	• Data type description	(Any) <b>whole number</b>	
	Cambridge2021		
	• Data type name	String	
• Data type description	A group of characters/text		
47.86			
• Data type name	Real		
• Data type description	(Any real) number that could be a whole number or a fraction		



Question	Answer	Marks
4(a)	<p><b>One mark per mark point (Max 3)</b></p> <p>MP1 Marks input are <b>stored in the array</b> <code>Score []</code></p> <p>MP2 Marks are checked against a range of boundaries // allow example</p> <p>MP3 ... and a <b>matching grade</b> is assigned to each mark that has been input</p> <p>MP4 ... then <b>stored in the array</b> <code>Grade []</code>...</p> <p>MP5 ... at the same index as the mark input</p> <p>MP6 The algorithm finishes after 30 marks have been input // allows 30 scores to be entered</p>	<b>3</b>
4(b)	<p><b>One mark per mark point (Max 3)</b></p> <p>MP1 Correct loop, including counter if not a FOR loop</p> <p>MP2 Correct output of <code>Score []</code></p> <p>MP3 Correct output of <code>Grade []</code></p> <p>MP4 Suitable messages/text in output for both arrays</p> <p>Example answers</p> <pre>Count ← 0 REPEAT   PRINT "Student: ", Count, " Mark: ", Score[Count], " Grade: ",Grade[Count]   Count ← Count + 1 UNTIL Count = 30  Count ← 0 WHILE Count &lt; 30 DO   PRINT "Student: ", Count, " Mark: ", Score[Count], " Grade: ",Grade[Count]   Count ← Count + 1 ENDWHILE  FOR Count ← 0 TO 29   PRINT "Student: ", Count, " Mark: ", Score[Count], " Grade: ", Grade[Count] NEXT</pre>	<b>3</b>

Question	Answer	Marks
4(c)	Any <b>three</b> correct statements ( <b>Max 3</b> ) e.g. MP1 Add an input facility to allow teachers to enter the class size MP2 Add a variable to store the input class size MP3 Use the class size variable as the terminating condition for the loop MP4 Make sure the arrays are sufficiently large to accommodate the largest possible class size	<b>3</b>

Question	Answer	Marks																																												
5(a)	<p><b>One</b> mark for each correct column (<b>Max 4</b>)</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Diff1</th> <th>Diff2</th> <th>OUTPUT</th> </tr> </thead> <tbody> <tr> <td>50</td> <td>50</td> <td>0</td> <td>Accept: Extreme</td> </tr> <tr> <td>75</td> <td>25</td> <td>25</td> <td>Accept: Normal</td> </tr> <tr> <td>99</td> <td>1</td> <td>49</td> <td>Accept: Normal</td> </tr> <tr> <td>28</td> <td></td> <td></td> <td>Reject: Abnormal</td> </tr> <tr> <td>82</td> <td>18</td> <td>32</td> <td>Accept: Normal</td> </tr> <tr> <td>150</td> <td></td> <td></td> <td>Reject: Abnormal</td> </tr> <tr> <td>-1</td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>	Value	Diff1	Diff2	OUTPUT	50	50	0	Accept: Extreme	75	25	25	Accept: Normal	99	1	49	Accept: Normal	28			Reject: Abnormal	82	18	32	Accept: Normal	150			Reject: Abnormal	-1																<b>4</b>
Value	Diff1	Diff2	OUTPUT																																											
50	50	0	Accept: Extreme																																											
75	25	25	Accept: Normal																																											
99	1	49	Accept: Normal																																											
28			Reject: Abnormal																																											
82	18	32	Accept: Normal																																											
150			Reject: Abnormal																																											
-1																																														

Question	Answer	Marks
5(b)	<p><b>One</b> mark per bullet point (<b>Max 2</b>)</p> <ul style="list-style-type: none"> <li>To output the type of test data</li> <li>... by performing a range check //... by checking if numbers are within the range 50 and 100 (inclusive) (or not).</li> </ul>	<b>2</b>

Question	Answer	Marks																																				
6(a)	The data in the ID column/field is unique/not repeated in each row/record	<b>1</b>																																				
6(b)	18	<b>1</b>																																				
6(c)	<table border="1" style="margin-left: 40px;"> <tbody> <tr> <td>Field:</td> <td>ID</td> <td>GenreName</td> <td>Overdue</td> <td></td> <td></td> </tr> <tr> <td>Table:</td> <td>GENRE</td> <td>GENRE</td> <td>GENRE</td> <td></td> <td></td> </tr> <tr> <td>Sort:</td> <td></td> <td></td> <td>Descending</td> <td></td> <td></td> </tr> <tr> <td>Show:</td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input checked="" type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> </tr> <tr> <td>Criteria:</td> <td></td> <td></td> <td>&gt;0</td> <td></td> <td></td> </tr> <tr> <td>or:</td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table> <p> <b>One</b> mark for the correct fields present and correctly named  <b>One</b> mark for correct table name and show box in all columns  <b>One</b> mark for correct sorting  <b>One</b> mark for correct search criterion         </p>	Field:	ID	GenreName	Overdue			Table:	GENRE	GENRE	GENRE			Sort:			Descending			Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Criteria:			>0			or:						<b>4</b>
Field:	ID	GenreName	Overdue																																			
Table:	GENRE	GENRE	GENRE																																			
Sort:			Descending																																			
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>																																	
Criteria:			>0																																			
or:																																						